Attorney Docket No. 42P14939 *PATENT*

UNITED STATES PATENT APPLICATION

FOR

METHOD AND APPARATUS FOR PERFORMING CYCLIC REDUNDANCY CHECKS

INVENTORS:

DONALD F. HOOPER
MATTHEW J. ADILETTA
STEPHANIE L. HIRNAK

PREPARED BY:

BLAKELY, SOKOLOFF, TAYLOR & ZAFMAN LLP
12400 WILSHIRE BOULEVARD
SEVENTH FLOOR
LOS ANGELES, CA 90025-1030

(408) 720-8300

EXPRESS MAIL CERTIFICATE OF MAILING

"Express Mail" mailing label number EV336586095US
Date of Deposit September 18, 2003
I hereby certify that I am causing this paper or fee to be deposited with the United States Postal Service "Express Mail Post Office to Addressee" service under 37 CFR 1.10 on the date indicated above and is addressed to Mail Stop Patent Application, Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.
Anne Collette
(Typed or printed name of person mailing paper or fee)
Anne Collette 9/18/2003
(Signature of person mailing paper or fee) Date

METHOD AND APPARATUS FOR PERFORMING CYCLIC REDUNDANCY CHECKS

FIELD OF THE DISCLOSURE

[0001] This disclosure relates generally to the field of data transmission. In particular, the disclosure relates to calculation and storage of cyclic redundancy check (CRC) residues with associated connection indices for multiple connections.

BACKGROUND OF THE DISCLOSURE

[0002] A cyclic redundancy check (CRC) is a widely used technique wherein a particular polynomial is used to assure data reliability. The CRC may be used to protect blocks of data called frames or cells wherein, the transmitter appends an extra n-bit sequence called a frame check sequence (FCS). The FCS holds redundant information about the frame that helps the receiver detect errors.

[0003] For example, some commonly used polynomials for CRC calculations are:

CRC-12: $X^{12}+X^{11}+X^3+X^2+X+1$,

CRC-16: $X^{16}+X^{15}+X^2+1$,

CRC-CCITT: $X^{16}+X^{12}+X^5+1$,

CRC-32: $X^{32}+X^{26}+X^{23}+X^{22}+X^{16}+X^{12}+X^{11}+X^{10}+X^8+X^7+X^5+X^4+X^2+X+1$.

[0004] CRC-32, for example, generates a 32-bit FCS, which is used by the local network standards committee (IEEE-802) and some Department of Defense (DOD) applications. The same algorithm or hardware may be used to compute the FCS or to check the data using the FCS. By appending an extra n-bit sequence of zeroes to the data as input, an FCS will be

produced as output. By appending the n-bit FCS sequence to the data as input, an n-bit sequence of zeroes will be produced as output if there are no transmission errors.

[0005] A network processor may perform CRC calculations as a transmitter and/or as a receiver for multiple frames or cells of one data packet or of multiple data packets. These packets may be of various different internetworking protocols or of one particular internetworking protocol. For example, Asynchronous Transfer Mode (ATM) is a set of protocols that use the concept of a connection and require ATM specific signaling protocols and addressing structures, as well as protocols to route ATM connection requests across an ATM network. A virtual circuit is set up across the ATM network prior to transferring data. ATM circuits can be virtual paths, identified by a virtual path identifier (VPI) or virtual channels, identified by the combination of a VPI and a virtual channel identifier (VCI). A virtual path is a bundle of virtual channels with a common VPI.

[0006] A typical network processor while operating as an ATM switch receives a cell of a specific VCI/VPI value; looks up the connection value in a local connection table to retrieve an associated CRC residue, identify an outgoing port and optionally get a new VPI/VCI value for that particular connection; calculates a new CRC residue for the connection to update the local connection table, and retransmits the cell on that outgoing link with appropriate connection identifiers. For example, one of the more common ATM Adaptation Layers (AAL) used to transmit data across ATM networks is called AAL5. AAL5 requires packets transmitted using a particular connection to be received in sequence, without interleaving cells of different packets on the same connection so that the receiver will be able to reconstruct the packet. Updating the CRC residue for the connection, permits sequential CRC calculations for the packet. Unfortunately, as the network processor may be

handling a large number of connections, the local table may become rather large, and latencies for repeated accesses to retrieve and to update CRC residues may represent a significant fraction of time in processing the packet.

[0007] Some network processors have provided for switching between multiple software threads of execution to permit access latencies to overlap with useful computations. Commonly though, cases may arise in which adjacent software threads may be processing sequential cells of the same connection as shown with reference to adjacent threads 301 and 302 of Figure 3. In such cases, an access 315 to retrieve an updated CRC residue must necessarily be performed sequentially with respect to the access 314 that updates the CRC residue. Therefore, potential throughput of the network processor may be limited by the frequency of occurrence of these cases. Unfortunately, the frequency may be rather high for some commonly used AALs (like AAL5 for example).

BRIEF DESCRIPTION OF THE DRAWINGS

[0008] The present invention is illustrated by way of example and not limitation in the figures of the accompanying drawings.

[0009] **Figure 1** illustrates one embodiment of a system for performing cyclic redundancy checks (CRC).

[0010] **Figure 2** illustrates a flow diagram for an exemplary process to transmit and receive data using Asynchronous Transfer Mode (ATM).

[0011] **Figure 3** illustrates a flow diagram for a prior art process for performing sequential CRCs with multiple threads.

[0012] **Figure 4** illustrates one embodiment of a process and apparatus using thread connection indices to perform CRCs.

[0013] **Figure 5a** illustrates a flow diagram for one embodiment of a process for performing CRCs using thread connection indices.

[0014] **Figure 5b** illustrates a flow diagram for an alternative embodiment of a process for performing CRCs using thread connection indices.

[0015] **Figure 6** illustrates an alternative embodiment of a process and apparatus using a content addressable memory to perform CRCs.

[0016] **Figure 7** illustrates a flow diagram for one embodiment of a process for performing CRCs using a content addressable memory.

DETAILED DESCRIPTION

[0017] Disclosed herein are processes and apparatuses for performing centralized cyclic redundancy checks (CRC). For one embodiment a current thread of execution compares a connection index with that of a previous thread of execution. If the two threads share the same connection index, a CRC calculation may be requested without providing a CRC residue to the centralized CRC unit since the CRC residue most recently produced by the CRC unit would have resulted from a preceding sequential cell of the same packet. Therefore the CRC calculation may proceed without suffering the latency associated with reading and writing the CRC residue.

[0018] For an alternative embodiment a current thread of execution requests a CRC calculation providing a connection index to the centralized CRC unit, which uses the connection index to access a content addressable memory (CAM). If the connection index produces a hit in the CAM, the CRC unit may use the CRC residue in the CAM associated with the connection index since it would have resulted from a preceding sequential cell of the same packet. Therefore the CRC calculation may proceed without suffering the latency associated with reading and writing the CRC residue even when threads processing sequential cells of the same connection are interleaved for execution with threads of a different connection.

[0019] These and other embodiments of the present invention may be realized in accordance with the following teachings and it should be evident that various modifications and changes may be made in the following teachings without departing from the broader spirit and scope of the invention. It will be appreciated that while examples presented below

illustrate performing centralized CRC calculations, for example in the context of Asynchronous Transfer Mode (ATM), the techniques disclosed are more broadly applicable. For example, statistics counters, Transmission Control Protocol (TCP) check-sum calculations or other similar calculations may make good use of the techniques herein disclosed to provide for increased overlap of access latencies and useful computations. The specification and drawings are, accordingly, to be regarded in an illustrative rather than restrictive sense and the invention measured only in terms of the claims and their equivalents.

[0020] **Figure 1** illustrates one embodiment of a system 101 for performing cyclic redundancy checks (CRCs). System 101 comprises one or more network interface devices 140 for transmitting and/or receiving data on a network 145, a serial dynamic random access memory (SDRAM) 120 to store the data and a network processor 102 to process the data, including performing CRC calculations.

[0021] Network processor 102 comprises an execution core 110, a system bus interface 111, a network bus interface 114 including a receive first-in-first-out (FIFO) buffer 142 and a transmit FIFO buffer 141 to buffer data received by or to be transmitted by the one or more network interface devices 140, and an SDRAM interface 112 to access data in receive FIFO buffer 142, transmit FIFO buffer 141 and/or SDRAM 120. A CRC circuit 121 is coupled with the SDRAM interface 112 to perform CRC calculations on data accessed in receive FIFO buffer 142, transmit FIFO buffer 141 and/or SDRAM 120 and to produce CRC residues.

[0022] A static random access memory (SRAM) 160, accessible via SRAM interface 116, stores instructions of threads of execution to be executed by micro-engine execution

circuits 131-136 to process data associated with virtual connections. Micro-engine execution circuits 131-136 execute these threads of execution, which may request CRC circuit 121 to perform CRC calculations on data in receive FIFO buffer 142, transmit FIFO buffer 141 and/or SDRAM 120 to produce CRC residues. One embodiment of network processor 102 optionally comprises an internal storage that stores data structures 115 accessed by micro-engine execution circuits 131-136 and/or by CRC circuit 121. Data structures 115 may include a virtual connection table 152 to hold, for example, VPI/VCI values, outgoing ports, CRC residues, etc.; and a thread connection array 151 to hold connection indices assigned to threads of execution; or optionally including a content addressable memory (CAM) 153 to hold CRC residues associated with particular connection indices. For alternative embodiments of network processor 102, all or some of data structures 115 may be stored in SRAM 160, or in SDRAM 120. For example, in one alternative embodiment of network processor 102, thread connection array 151 or CAM 153 may be stored internally while virtual connection table 152 may be stored in SRAM 160.

[0023] **Figure 2** illustrates a flow diagram for exemplary process 201 to transmit and exemplary process 202 to receive encapsulated Ethernet data using Asynchronous Transfer Mode (ATM). Process 201, process 202 and other processes herein disclosed are performed by processing blocks that may comprise dedicated hardware or software or firmware operation codes executable by general purpose machines or by special purpose machines or by a combination of both. It will be appreciated that while process 201, process 202 and other processes herein disclosed are illustrated, for the purpose of clarity, as processing blocks with a particular sequence, some operations of these processing blocks may also be conveniently performed in parallel or their sequence may be conveniently permuted so that

the some operations are performed in different orders, or some operations may be conveniently performed out of order.

[0024] In processing block 210, data is received from an Ethernet transmission. The data is enqueued for CRC calculations in processing block 211. Payload data for each cell may be buffered in SDRAM 120, for example. Processing then proceeds in processing block 212 where the AAL5 CRC residues are generated. It will be appreciated that it is not necessary that data from all of the cells is enqueued before processing of AAL5 CRC residues may proceed in processing block 212, CRC residues being carried forward to each successive cell. As cells are processed they are assembled into the packet for transmission and the AAL5 CRC residue of the last cell of the packet is affixed to the packet. Then, in processing block 213, data packets and CRC residues for a particular connection are transmitted using ATM.

[0025] In processing block 214, packets and CRC residues for a particular connection are received from ATM transmission. The data cells are enqueued for CRC checks in processing block 215. Processing then proceeds in processing block 216 where the AAL5 CRC residues are used to check the data transmission, beginning with the residue that was affixed to the packet and with residues again being carried forward to each successive cell. The AAL5 CRC residue of the last cell of the packet should result in zeroes if there are no errors. Then, in processing block 217, the data that has been reassembled is ready for Ethernet transmission.

[0026] **Figure 3** illustrates a flow diagram for a prior art process for performing sequential CRCs with multiple threads. Data packets and CRC residues are received and

transmitted for connection 346 and connection 347 over the network 345 by network interface 340. Cells from the packets are buffered by buffers 342.

[0027] One or more execution circuits may execute thread 301 and thread 302, which may request CRC circuit 321 to perform CRC calculations on the data cells in buffers 342 to produce CRC residues. A virtual connection (VC) table 352 holds CRC residues for each of connection 346 and connection 347. In the example illustrated, thread 301 and thread 302 are executed adjacently and thread 301 and thread 302 process sequential cells of the same connection. Therefore, as will be further described, an access 315 to retrieve a CRC residue from VC table 352 must be performed sequentially with respect to the access 314 that updates the CRC residue in VC table 352.

[0028] Responsive to the execution of thread 301, processing block 311 reads a residue for a particular connection from VC table 352. Horizontal dotted arrows illustrated between processing blocks represent some nonspecific time before the beginning of a subsequent processing block, which is usually associated with the latencies of accessing VC table 352 or CRC circuit 321. In processing block 312, the residue retrieved from VC table 352 is written to CRC circuit 321 and a CRC calculation is initiated and subsequently performed by CRC circuit on a cell from buffers 342 of the connection associated with thread 301. Processing block 313 reads the CRC residue produced by CRC circuit 321 in response to the CRC calculation initiated in processing block 312. In processing block 314, the CRC residue that was read in processing block 313 is written to VC table 352 to update the CRC residue for the connection associated with thread 301. A signal may also be generated to indicate that CRC processing of the next sequential cell of the connection may begin.

[0029] Since thread 301 and thread 302 process sequential cells of the same connection, execution of thread 302 may now read the updated CRC residue. Responsive to the execution of thread 302, processing block 315 reads the updated residue for the connection from VC table 352 and subsequently goes on to perform a CRC calculation substantially similar to that of thread 301. In such cases as described with respect to Figure 3, potential throughput of a network processor may be limited whenever a read access to retrieve a CRC residue from VC table 352 must be performed sequentially with respect to the write access that updates the CRC residue in VC table 352.

[0030] **Figure 4** illustrates one embodiment of a process and apparatus using thread connection indices to perform CRC calculations, which may provide for improved throughput of a network processor for processing sequential cells of the same connection. Data packets and CRC residues are received and transmitted for connection 446 and connection 447 over the network 445 by network interface 440. Cells from the packets are buffered by buffers 442. For one embodiment of network processor 102, buffers 442 may reside in network bus interface 114. For an alternative embodiment of network processor 102, buffers 442 may reside in SDRAM 120 or in SDRAM interface 112.

[0031] VC table 452 holds CRC residues for all connections including connection 446 and connection 447. Thread connection array 451 stores entries 452-456 for all threads, Each of entries 452-456 associates a thread's Id (or number) with a connection index for performing CRC calculations, for example, using the CRC residue produced for previous thread 401 when thread 402 and previous thread 401 share the same connection index.

[0032] In the example illustrated, thread 401 and thread 402 are executed adjacently and thread 401 and thread 402 process sequential cells of the same connection. One or more

execution circuits may execute thread 401 and thread 402, which may request CRC circuit 421 to perform CRC calculations on data cells in buffers 442 to produce CRC residues.

[0033] Responsive to the execution of thread 401, processing block 411 gets its connection index. One embodiment of a connection index may be assigned to a thread by a hash lookup using the VCI/VPI value of the connection. One embodiment might assign to a thread the VC table 452 entry of the connection as a connection index. An alternative embodiment might assign to a thread a pointer to the entry in the VC table 452. A signal may also be generated to indicate that CRC processing of the next sequential cell of the connection or of the next thread may begin. For one embodiment of network processor 102, threads may be scheduled in round-robin order. For alternative embodiments, some other order for scheduling may be conveniently chosen. Since thread 401 and thread 402 are adjacent threads and also process sequential cells of the same connection, execution of thread 402 may now begin with processing block 414 which gets its connection index.

[0034] Processing block 412 writes its connection index to TC array 451, reads the connection index of the previous thread from TC array 451, and reads a residue for its connection from VC table 452, some or all of these accesses which may be performed concurrently or in parallel. A signal may also be generated to indicate that CRC processing of the next thread may continue. Subsequently, processing block 415 writes its connection index to TC array 451, reads the connection index of previous thread 401 from TC array 451, and reads a residue for its connection from VC table 452.

[0035] In processing block 413 of thread 401, the residue retrieved from VC table 452 is written to CRC circuit 421 (if the connection index of the thread previous to thread 401 was different than the connection index of thread 401) and a CRC calculation is initiated and

subsequently performed by CRC circuit on a cell from buffers 442 of the connection associated with thread 401. Again a signal may also be generated to indicate that CRC processing of the next thread may continue. In processing block 416 of thread 402, the residue retrieved from VC table 452 is not written to CRC circuit 421 because the connection index of thread 401, which is previous to thread 402 is the same connection index of thread 402. Therefore the CRC residue retrieved from VC table 452 is stale and the residue most recently produced by CRC circuit 421 for thread 401 and already stored in CRC circuit 421 is the correct residue, so a CRC calculation is initiated in processing block 416 and subsequently performed by CRC circuit 421 on a cell from buffers 442 of the same connection associated with thread 401 and thread 402. Thus use of thread connection indices in TC array 451 may provide for an increased overlap of thread 401 and thread 402 execution, reduced sequential latencies, and improved throughput of a network processor for processing sequential cells of the same connection.

[0036] Further processing (not shown) in thread 401 and thread 402 continues to read their respective CRC residues produced by CRC circuit 421 and to write their respective CRC residues to VC table 352 to update the CRC residue for the connection associated with thread 401 and thread 402. It will be appreciated that the above example may be extended for alternative embodiments including multiple CRC circuits and further including comparison of connection indices for other threads in addition to the most previous thread.

[0037] **Figure 5a** illustrates a flow diagram for one embodiment of a process 501 for performing CRCs using thread connection indices. In processing block 511, a lookup of the current thread's connection index is performed. In processing block 512, the current thread writes its connection index to thread connection array (TCA) 151. In processing block 513,

the current thread reads the connection index of the previous thread from TCA 151. In processing block 514, the connection index of the current thread is compared to connection index of the previous thread. If they are different, processing continues in processing block 515 where the current thread reads a residue for its connection from VC table 152, and in processing block 516 where the current thread writes the residue to CRC circuit 121.

[0038] Otherwise, if the connection index of the current thread is equal to the connection index of the previous thread in processing block 514, processing proceeds directly to processing block 517 where a CRC calculation is requested from CRC circuit 121. Processing then continues in processing block 518, where the current thread reads the CRC residue from CRC circuit 121, and in processing block 519, where the current thread writes the CRC residue to VC table 152.

[0039] As previously suggested, one or more operations of the processing blocks 511-519 may be conveniently performed concurrently or in parallel or in different orders (for example, processing blocks 512-513).

[0040] **Figure 5b** illustrates a flow diagram for an alternative embodiment of a process for performing CRCs using thread connection indices. In processing block 511, a lookup of the current thread's connection index is performed.

[0041] In processing block 512, the current thread writes its connection index to TCA 151. In processing block 513, the current thread reads the connection index of the previous thread from TCA 151. In processing block 515, the current thread reads a residue for its connection from VC table 152. As previously suggested, some or all of the accesses of processing block 522 may be performed concurrently of in parallel, thereby overlapping the associated latencies.

[0042] In processing block 514, the connection index of the current thread is compared to connection index of the previous thread. If they are different, processing continues in processing block 516 where the current thread writes the residue to CRC circuit 121.

[0043] Otherwise, if the connection index of the current thread is equal to the connection index of the previous thread in processing block 514, processing proceeds directly to processing block 517 where a CRC calculation is requested from CRC circuit 121. Processing then continues in processing block 518, where the current thread reads the CRC residue from CRC circuit 121, and in processing block 519, where the current thread writes the CRC residue to VC table 152.

[0044] **Figure 6** illustrates an alternative embodiment of a process and apparatus using a content addressable memory to perform CRCs, which may provide for improved throughput of a network processor for processing cells of the same connection. Data packets and CRC residues are received and transmitted for connection 646 and connection 647 over the network 645 by network interface 640. Cells from the packets are buffered by buffers 642.

[0045] VC table 652 holds CRC residues for all connections including connection 646 and connection 647. Content addressable memory (CAM) 653 stores entries 655, 657... 659 for active connections having associated threads that have provided connection indices when requesting CRC calculations. These connection indices are associated with the last CRC residue produced for a thread of the same connection index.

[0046] One or more execution circuits may execute thread 601 and thread 602, which may request CRC circuit 621 to perform CRC calculations on data cells in buffers 642 to produce CRC residues.

[0047] Processing (not shown) in thread 601 and thread 602 begin with getting their threads' respective connection indices substantially similar to the processes described above. Further responsive to the execution of thread 601, processing block 611 reads a residue for its connection from VC table 652. A signal may also be generated to indicate that CRC processing of the next thread may continue. Subsequently, processing block 614 reads a residue for its connection from VC table 652.

[0048] In processing block 612 of thread 601, the thread's connection index and the residue retrieved from VC table 652 is supplied to CRC circuit 621 and a CRC calculation is initiated. CRC circuit 621 accesses CAM 653 using the thread's connection index. A CRC calculation using the residue retrieved from VC table 652 in case of a miss in CAM 653 and using the CRC residue associated with the thread's connection index in case of a hit in CAM 653 is subsequently performed by CRC circuit 621 on a cell from buffers 642 of the connection associated with thread 601. Again a signal may also be generated to indicate that CRC processing of the next thread may continue.

[0049] In processing block 615 of thread 602, the thread's connection index and the residue retrieved from VC table 652 is supplied to CRC circuit 621 and another CRC calculation is initiated. CRC circuit 621 again accesses CAM 653 using the connection index of thread 602. A CRC calculation, using the residue retrieved from VC table 652 in case of a miss in CAM 653 and using the CRC residue associated with the thread's connection index in case of a hit in CAM 653, is subsequently performed by CRC circuit 621 on a cell from buffers 642 of the connection associated with thread 601. Thus use of thread connection indices in CAM 653 may provide for an increased overlap of thread 601

and thread 602 execution, reduced sequential latencies, and improved throughput of a network processor for processing cells of the same connection.

[0050] Processing block 613 in thread 601, and processing block 616 in thread 602 continue to get the respective CRC residues produced by CRC circuit 621 and to write their respective CRC residues to VC table 652.

[0051] **Figure 7** illustrates a flow diagram for one embodiment of a process for performing CRCs using a content addressable memory. In processing block 711, a lookup of the current thread's connection index is performed. In processing block 715, the current thread reads a residue for its connection from VC table 652. Processing continues in processing block 716 where the current thread provides the residue and the current thread's connection index to CRC circuit 621. CRC circuit 621 accesses CAM 653 using the current thread's connection index and replaces the residue with the residue retrieved from VC table 652 in case of a miss in CAM 653. Otherwise, in case of a hit in CAM 653, CRC circuit 621 replaces the residue with the CRC residue in CAM 653 associated with the current thread's connection index. Therefore the CRC residue retrieved from VC table 652 is stale and the residue most recently produced by CRC circuit 621 for the current connection index and already stored in CAM 653 is the correct residue.

[0052] Processing proceeds to processing block 717 where a CRC calculation is requested from CRC circuit 621. Subsequently a CRC calculation is performed by CRC circuit 621 on a cell of the connection associated with the current thread and in processing block 753, the resulting CRC residue is stored in CAM 653. Processing then continues in processing block 718, where the current thread reads the CRC residue from CRC circuit 621,

and in processing block 719, where the current thread writes the CRC residue to VC table 652.

[0053] The above description is intended to illustrate preferred embodiments of the present invention. From the discussion above it should also be apparent that especially in such an area of technology, where growth is fast and further advancements are not easily foreseen, the invention may be modified in arrangement and detail by those skilled in the art without departing from the principles of the present invention within the scope of the accompanying claims.